

08/818053



Attorney Docket No.: 0548-VDSK

Patent Application

ASSISTANT COMMISSIONER FOR PATENTS
 Washington, D.C. 20231
 BOX PATENT APPLICATION - FEE

SIR: Transmitted herewith for filing is the utility patent application of
 Inventor(s): Goran DEVIC
 For: METHOD AND APPARATUS FOR SHORTENING DISPLAY LIST INSTRUCTIONS

Enclosed are:

- XX 4 sheets of Drawings with 5 figures.
- XX 29 sheets of specification.
- XX A signed Declaration.
- XX An assignment of all rights to Cirrus Logic, Inc. (with coverage).

The Filing Fee has been calculated as shown below:

	NO. OF CLAIMS		EXTRA CLAIMS	RATE	FEE
Basic Application Fee					\$770.00
Total Claims	30	Minus 20=	10	X \$22	\$220.00
Independent Claims	3	Minus 3=	0	X \$80	\$0.00
If multiple claims newly presented, add \$260					\$0.00
TOTAL FEE DUE					\$990.00


The Commissioner is hereby authorized to charge payment of the following fees associated with this communication, or credit any overpayment, to our Deposit Account No. 03-2028. any filing fees required under 37 CFR Section 1.16; any patent application processing fees under 37 CFR Section 1.17 and any additional fees which might be required during the pendency of this application. Form PTO/SB/17 has been provided for deposit account charging purposes.

Please send all correspondence to the undersigned at:

Cirrus Logic, Inc.
 3100 West Warren Avenue, MS 521
 Fremont, CA 94538-6423
 (510) 252-6242
 FAX (510) 252-6230.

Date: March 14, 1997

Respectfully Submitted,
 CIRRUS LOGIC, INC.


 Steven A. Shaw
 Attorney for Applicants
 Reg. No. 39,368

EXPRESS MAIL CERTIFICATE OF MAILING 37 CFR 1.10
 "Express Mail" mailing label number EM283262400US
 Date of Deposit March 14, 1997
 I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Print Name

Signature

Steven A. Shaw
3/14/97
 Date

EM283262400US



United States Patent Application

**METHOD AND APPARATUS FOR SHORTENING DISPLAY LIST
INSTRUCTIONS**

Inventor: GORAN DEVIC
10401 Doc Holliday Trail
Austin, TX 78753

I hereby certify that this document is being deposited with the United States Postal Service as Express Mail in an envelope
marked as "Express Mail Post Office to Addressee" Mailing
Label Number PM2 83262400US
addressed to the Commissioner of Patents and Trademarks,
Washington, D.C. 20231 on this date March 14, 1997

Steven A. Shaw
Typed or printed name of person mailing this document

Signature

Date

Assignee: **Cirrus Logic, Inc.**
3100 West Warren Ave.
MS 521; Legal/IP
Fremont, CA 94538
(510) 252-6242

08618053-034497

Field of Invention

The present invention relates generally to a graphics system for personal computers. More particularly, the present invention relates to a method and apparatus for shortening display list instructions in a graphics processor.

Description of the Related Art

Sophisticated graphics packages have been used for some time in expensive computer design and graphics systems. Increased capabilities of graphics controllers and display systems, combined with standardized graphics languages, have made complex graphics functions available in even the most routine applications. For example; word processor, spread sheets and desktop publishing packages now include relatively sophisticated graphics capabilities. Three-dimensional (3D) displays have become common in games, animation, and multimedia communication and drawing packages.

The availability of sophisticated graphics in PCs has driven a demand for even greater graphics capabilities. To obtain these capabilities, graphics systems must be capable of performing more sophisticated functions in less time to process greater amounts of graphical data required by modern software applications. In particular, there is a continuing need for improvements in software algorithms and hardware implementations to draw three-dimensional objects using full color, texture mapping and transparency blending.

Improvements have been made in the hardware realm. Graphics processors and accelerators are available with software drivers that interface with a host central processing unit to the graphics processor. In general, the software receives information for drawing objects on a computer screen, calculates certain basic parameters associated with the objects and provides this to the graphics processor in the form of a "display list" of parameters. A graphics controller then uses the display list values in generating the graphics to be displayed. A graphics processor may use interpolation techniques where the fundamental information for the object to be drawn comprises a series of initial and incremental parameters or values. The graphics processor loads or otherwise receives the initial parameters for the pixels to be drawn, interpolate the object by incrementing the parameters until the object is completely drawn.

In many prior art computer systems, external devices such as graphics devices are able to read a stream of data (display list) from memory and execute programs stored in the memory in a similar manner. The size of these display list information tend to place limitations on the traversal (read/write) speed of the central processing unit and the graphics processor.

The CPU typically builds the display list information with the instructions and parameters specific to the particular external device attached to the computer system. The external device then reads the instruction stream and executes instructions from this stream. One of the common operations stored in the display list is

a command to load single and multiple registers of a device's register file with specified values.

Existing graphics implementation that use display lists typically load data in a sequential format to a register file in the graphics processor. For each type of primitive, a particular set of data values are required to render that type of primitive. For example, a point to be drawn to a pixel grid requires an X,Y location, color values and a Z value for depth comparison. An example of display list is shown below in Table I.

<u>ADDRESS</u>	<u>NAME</u>	<u>DESCRIPTION</u>
0x4000	X	Initial X value
0x4004	Y	Initial Y value
0x4008	Z	Initial Z value
0x400C	R	Initial Red component
0x4010	G	Initial Green component
0x4014	B	Initial Blue component
0x4018	X1	Some other register
0x401C	X2	- -
0x4020	A	Alpha blending value

TABLE I

The display list in Table I provides the parameters required to draw points, lines and polygons. From the display list provided above, if a specific primitive rendering operation requires, for example, only the following register values to be loaded (e.g.,

X,Y,R,G,B and A); a prior art load instruction would use one of two alternative methods of instruction loading.

The first of the two alternatives will be to load all nine registers (e.g., "Load instruction (start at X),

5 X,Y,Z,R,G,B,X1,X2,A"). The stream of information in the display list will therefore occupy 10 instruction words (40 bytes) and load unnecessary registers.

The second load alternative is to use two consecutive load operations thereby replacing the two register load gaps (e.g.,

10 X1,X2) with only one load instruction (e.g., "Load instruction (start at X), X,Y,Z,R,G,B" and "Load instruction (starts at A),

A"). The stream of information in the display list for this load sequence is 9 instruction words long (36 bytes). These two prior art instruction load methods have the common feature of

15 sequentially loading the register file with the parameter values for the primitive being rendered. Also, the load instructions comprise of two fields; a first field which holds the starting parameter value and a second field which holds the incremental count of subsequent parameter values for the primitive being

20 rendered.

Despite these prior methods instruction load operations and the ability to load multiple registers contiguously to enable the efficient processing of the display list, several problems emerge when the size of the display list gets too large.

25 One of such problems is that extra system memory may be needed to store the large display list. This may impose extra cost in the

overall price of the computer system. Although memory prices are getting a bit cheaper, the average amount of memory installed in many of today's multimedia computer systems continue to substantially increase. For example, a Pentium® based multimedia computer system running MS Windows® NT may require about at least 32 megabytes of memory to run efficiently.

As the memory requirements of these multimedia systems continue to grow, the memory required to maintain and execute very long display lists needed by the multiprogramming operating systems in these computer systems become very significant. Moreover, since the memory in these systems may become locked, (i.e., the operating system is not able to swap processing to the computer system's external storage device). Such a lock further reduces the amount of memory that is left for the computer system to process other system activities.

Another problem with the presence of long display list is the time needed by the CPU to build the list and for the external device to execute the list. If a high frame rate and fast response time is needed by the CPU, the time spent managing the display list must be minimized. The amount of information that is being transferred between the CPU and the external device should not be sacrificed since the approach would definitely affect the quality of the image being rendered. Even if the setting other than the computer graphics, the amount of information may have to be the same since the external device may need it all.

As more and more of the computer's processing power is transferred to the central processing unit, the processing of long display lists to generate graphics display end up being bottlenecks in processing instructions by the CPU. This problem becomes even
5 more pronounce if the processing of graphics data is transferred from a separate graphics processing chip of device to the CPU.

Thus, a method of shortening display list information without losing the quality of the information being passed, while maintaining the processing speed of CPU is needed. The present
10 invention provides the advantageous functionality of shortening display list information and the ability to randomly load register file in graphics processing device with a single load instruction.

Summary of the Invention

A method and apparatus are described herein which reduce processing time while maintaining the quality of display information and without requiring extra system memory. In accordance with the present invention, a graphics processor for generating shorter display list instructions without losing the quality of the display information supplied to a display screen is disclosed. The graphics processor provides a field load instruction which is generated by a central processing unit which is supplied to the graphics processor. the field load instruction is then encoded into the display list instruction for subsequent execution by a external graphics device in a computer system. By providing a short display list, the present invention provides a system which is able to handle the increasing amount of graphics data processed in many present day multimedia computer systems, without requiring excessive amount of memory resources.

Another embodiment is a computer controlled graphics display system having a processor coupled to a bus, a memory unit coupled to the bus for storing the display list, a graphics processor for receiving microinstructions from the display list stored in the memory unit, a set of register files coupled to the graphics processor for storing the shortened display list in the graphics processor, and a private memory area disposed within the memory unit for storing address offsets of the display list; wherein named instructions generated by the central processor replace other means of randomly loading the register file in the graphics processor.

Embodiments further include the above; wherein the display list comprises parameterization procedures for processing polygon primitives, sets of graphics lines, and sets of graphics points; and wherein the parameterization procedure are further for
5 processing translation between different graphics formats.

Embodiments further include the above; wherein the load instruction comprises instruction bit-field for performing specific instructions by the display list.

Embodiment further include the above; wherein the load
10 instruction further comprises an opcode bit-field for storing data representing opcode instruction in the display list.

Embodiments further include the above; wherein the load instruction further comprise an partition bit-field for storing partition data defining the partition index of the display list to
15 the private memory area.

The graphics processor also preferably includes an internal instruction execution unit that receives the opcode from a prefetch unit and decodes the opcode. The execution unit also receives the display list and stores the display list in a register file.

Brief Description of the Drawings

Figure 1 is a simplified block diagram of a graphics processor coupled to a system bus of a computer system, in accordance with the principles of the present invention.

5 Figure 2 is a simplified block diagram showing in more detail
a portion of the graphics subsystem of Figure 1.

Figure 3A is a simplified block diagram of the field load instruction unit Figure 2.

Figure 3B is a simplified block diagram of the partition look-
up table of Figure 2.

Figure 4 is a flow diagram of the display list shortening process of the present invention.

[illegible]

DETAIL DESCRIPTION OF THE PRESENT INVENTION

A method and apparatus for providing shorter display lists without losing the quality of the display information supplied to the graphics device is disclosed.

5 In the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without these specific details
10 or by using alternate elements or methods. In other instances well know methods, procedures, components, and circuits have been described in detail as not to unnecessarily obscure aspects of the present invention.

15 Some portions of the detailed description which follow are represented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer system. These descriptions and representations are the means used by those skilled in the art to most effectively convey the substance of their work to other skilled in the art. A
20 procedure, logic block, process etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, magnetic signals capable of being stored, transferred, combined, compared,
25 and otherwise manipulated in a computer system. For reasons of convenience, and with reference to common usage, these signals are

referred to as bits, values or the like with reference to the present invention.

It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that thorough discussions of the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computer system's registers and memories and is transformed into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

With reference to Figure 1, a block diagram is shown of a host computer system 100 used by the preferred embodiment of the present invention. In general, host computer 100 comprises a bus 101 for communicating data and instructions, a host processor (CPU) 102 coupled to bus 101 for processing data and instructions, a computer readable non-volatile memory unit 103 coupled to bus 101 for storing data and instructions from the host processor 102, a computer readable data storage device 104 coupled to bus 101 for storing data and display device 106 coupled to bus 101 for

displaying information to the computer user. The display device 106 utilized with the computer system 100 of the present invention can be a liquid crystal device, cathode ray tube, or other display device suitable for creating graphics images and alphanumeric characters recognizable to the computer user.

The host system 100 provides data and control signals via bus 101 to a graphics hardware subsystem 109. The graphics hardware 109 includes a display processor 110 which executes a series of display instructions found within a display list. The graphics display processor 110 supplies data and control signals to a frame buffer which refreshes the display device for rendering images on display device. Alternatively, the host processor 102 may write the display list to the graphics processor 110 in accordance with known techniques.

It should be understood that the particular embodiment shown in Figure 1 is only one of many possible implementations of a graphics system for use in a computer system. Figure 1 is simplified for purposes of clarity so that many components and control signals are omitted which are not necessary to understand the present invention.

In the preferred embodiment, the graphics processor 110 provides hardware support for 2D and 3D graphics, and for text and windowing operations of a computer system. The graphics processor 110 transfers digital data from the system memory 104 or host processor 102, and processes data for storage in the RDRAM 115 ultimately for display on the display unit 106.

In accordance with the preferred embodiment, the host processor 102 provides necessary parameter values in the form of a display list, which typically is stored in system memory 104 until required by graphics processor 110.

5 The host processor 102 and system memory 104 both preferably communicate with the graphics processor 110 via the system bus 101. The system bus 101 may comprise any one a plurality of different types of host or input/output (I/O) buses, including the industry standard architecture (ISA), the extended ISA (EISA), the
10 peripheral component interconnect (PCI) and any other standardized system bus of a computer system.

Still referring to Figure 1, the graphics processor 110 couples to the system bus 101. In accordance with the preferred embodiment, the graphics processor 110 preferably includes bus
15 mastering capabilities, thus permitting graphics processor 110 to bus master the system bus 101. Graphics processor 110 also couples to a display unit and a RDRAM 115. In the preferred embodiment, the RDRAM comprises a bank of RDRAM buffers, where the digital data stored in the RDRAM comprises a rectangular array of picture
20 elements referred to as pixels or pixel values. Each pixel can be defined by an 8 bit value, for example, which specifies the intensity of a single color of a corresponding pixel on a screen of the display unit 106.

The graphics device 109 hosts an array of volatile memory unit
25 referred to as register file 112. The register file 112 holds working information of the graphics device. The register file also

stores information and commands needed for operation of the graphics device 109.

The display unit 106 may be any suitable type of display device, such as a cathode ray tube (CRT) for desktop, workstation
5 or server applications, a liquid crystal display (LCD) or any other suitable display device for a personal computer.

The RDRAM frame buffer provides a performance improvement by permitting faster access to display list instructions and pixel data, compared to accessing data stored in the main memory 104 of
10 the host computer system 100. The graphics processor 110 communicates to the RDRAM buffer 115 through address data and control lines, collectively referred to as a RBUS 118.

Referring now to Figure 2, the graphics subsystem 109 preferably includes a register file 112, a graphics processor 110
15 and a frame buffer 115. Generally the register files 112 comprises a plurality of registers for storing the display list information. The register address generator generates the address pertaining to a register being accessed for display list information to be displayed.

20 The graphics processor 110 comprises a fetch subsequent parameters unit 200, a load instruction unit 210, a "right to left" shifter unit 220, an address counter 230 and a partition look-up table unit 240.

The field load instruction unit 210 comprises a plurality data
25 bit locations for storing load bit data for performing the display list load instruction in the graphics processor. A detailed

description of the field load instruction is given in Figure 3A below.

The fetch subsequent instruction parameter unit 200 is coupled to the register files 112, and operates to fetch subsequent display list instructions after a first instruction has been processed.

The fetch subsequent parameters unit is activated by the assertion of request for next parameter lines 201 by the graphics processor, 110. When the fetch subsequent parameter 200 detects that request for the next parameter lines 201 have been asserted, display list data is driven on data line 221 to the register file 112 for subsequent write operation to the CPU.

Field Load instruction unit 210 is coupled to shifter 220 to pass load instructions to the register file 112. Field load instruction unit 210 comprises a plurality of data bits of a specified value each of which defines an operation to be performed by the graphics processor 110 in processing the display list. The field load instruction unit 210 passes data to shifter 220 when write enable signal lines 211 are asserted.

The Write Enable signal lines 211 are assumed to be the topmost bit position in shifter 220. At each internal clock cycle of the graphics processor 110, the Write Enable signal 211 is propagated to the register file 112 and to the subsequent parameter fetch unit 200 to fetch subsequent graphics parameters.

If the Write Enable signal 211 is asserted (i.e., having a bit value of "1'), the register file 112 stores the data provided by the fetch subsequent parameter unit 200 in a register address

provided by the address generation unit via the address counter 230.

If the Write Enable signal 211 is reset (i.e., having a binary value of "0"), all writes to the register file 112 are disabled and the subsequent parameter fetch unit 200 fetches new parameters from the display list. The shifter 220 shifts its contents one (1) bit to the left following either a write enable or a write disable operation to the register file 112. Shifting bits in the shifter 220, in this manner, allows the next bit of a Write Enable operation to generate a write/skip signal to the register file 112. Consequently, the register files 112 is randomly loaded depending on whether the write enable data bit is set or not.

Address counter 230 is coupled to the register file 112 and the address generation unit 235 to incrementally load new request addresses to the register file 112. The address counter 230 continues to generate new addresses to the register file 112 until the field load instruction contained in a display list are completely executed.

Still referring to Figure 2, partition look-up table 240 comprises a plurality of preloaded addresses which offset into the register file 112. The partition look-up table 240 is loaded with new address after each display list has been completely processed by the graphics processor 110. The partition table is coupled to the field load instruction unit. Portions of the field load instruction unit 210 reference the contents of the partition table 240.

In the preferred embodiment of the present invention, the partition lookup table 240 comprises 64 entries each of which is addressed by a partition data bit in the load instructions.

In its basic implementation, the look-up table 240 contains
 5 the addresses of 64 registers which are evenly distributed across the 1024 register set of the register files 112. Thus, each field load instruction only needs 6 bits to specify the starting partition of the register file 112 to load the display list thereby shortening the display list. The field load instruction also
 10 allows the register files 112 to be randomly loaded.

Referring to Figure 3A is a simplified block diagram of a load instruction of the preferred embodiment. The load instruction shown in Figure 3A comprises an opcode field 300, a write enable field 310 and a partition field 320.

15 The field load instruction of the preferred embodiment can load all, and only the registers required by a display list. The instruction stream of an exemplary load instruction looks as follows: "Field Load (write enables: 110111001), (partition starts at X), X,Y,R,G,B,A". This data stream, unlike the prior art, is
 20 only 7 instructions word long (28bytes). The write enable field 310 contained in the load instruction, which read from left to right, allows writes (binary 1s) only on desired registers of the register files 112. The registers that are not to be set are skipped.

Still referring to Figure 3A, the opcode field 300 stores data
 25 of a distinctive bit pattern which recognizes the "field load" instruction from other instructions in the display list information

by the graphics processor 110. In the preferred embodiment of the present invention, the opcode is kept short to leave more space for the "write enable" and the "partition" field respectively.

5 The write enable field 310 stores data bits which may be set to enable or disable register write operations of the load instruction to the register files 112. In the present invention, the setting of the write enable bit-field allows the register files 112 to be randomly loaded with the display parameter values. For example, if the write enable bit-field in a particular load
10 instruction is enabled, the corresponding register location in the register files 112 is loaded with the display parameters.

Alternatively, if the write enable bit-field 310 is disabled, the write to the register files 112 will be disabled and the circuit which fetches subsequent parameters will request a next
15 parameter fetch from the display list. Consequently, the corresponding register position is skipped in the register files 112. Thus, depending on the contents of the write enable bit-field position in a load instruction, corresponding register locations may be written or skipped.

20 The partition bit-field portion of the load instruction stores data bits which indexes to the partition look-up table.

Figure 3B is a simplified block diagram illustrating an exemplary embodiment of the partition look-up table of the present invention. The partition look-up table 240 shown in Figure 3B
25 comprises of 64 entries of preloaded address offsets to the

register files 112. In the preferred embodiment, register files 112 comprises 1024 entries.

In order to address a particular register in the register files 112, prior art methods of addressing needed 10 binary bits of data to load each register. In the present implementation of the load instruction, partition look-up table 240 allows the register files 112 to be addressed with only 6 bits of data. The 64 entries in the partition look-up table 240 are evenly distributed across the register files 112 as shown in Figure 3B.

Referring to Figure 4, is a simplified block a flow process of the preferred embodiment of the present invention. The diagram shown in Figure 4 illustrates the execution of the "Field load" instruction. First at step 410, shifter 220 is loaded with the Write Enable data from the Write Enable field of the load instruction.

At step 420, the partition table is indexed using the partition instruction data bit from the load instruction. A base address of the first register in the register file 112 is then retrieved from the partition table 240 and loaded into the address counter 230 at step 430.

At step 440, the top bit of the shifter 220 is examined to determine whether the addressed register must be loaded or not. If the top-bit in shifter 220 is set, then the subsequent parameter fetch unit fetches the next parameter from the display list at step 450 and stores the retrieved data in the register file 112.

If the top-bit in shifter 220 is not set, then the address counter 230 increments the address count at step 460.

At step 470, shift register 220 is shifted one bit to the left after address counter 230 has been increased 1 bit. After the shifter has been shifted a bit, the contents of shifter 220 are examined to determine if it is empty at step 480. If the shifter 220 is empty processing of a current display list end at step 490. If, on the other hand, there is more data in the shifter 220, the graphics processor 110 continues to execute the current display list at step 440. The effect of loading the shifter 220 and incrementing the address count is to effectively load multiple register in the register file 112 randomly at once. Random loading of the register files 112 in this manner effectively shortens the display list compared to the traditional way of loading instructions in a sequential or contiguous manner.

Thus a method and an apparatus for shortening display list instruction through a random loading of register files is disclosed. The preferred embodiment of the present invention is described for illustrative purposes, numerous other variations of the disclosed embodiments will be apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such modification and variations.

00010000-00000000

CLAIMS

What is claimed is:

1 1. A system for rendering graphics primitives using a shortened
2 display list for ensuring a shortened processing time while
3 maintaining the quality of the display information contained in the
4 display list, the system comprising:

5 a system bus for communicating data and instructions;

6 a host processor coupled to the system bus for processing a
7 display list defining a graphics primitive;

8 a system memory coupled to the system bus for storing the
9 display list;

10 a graphics subsystem coupled to the system bus for processing
11 display parameter values contained in the display list, wherein the
12 display list includes a field load instruction for effectively
13 shortening display list processing; and

14 a display unit coupled to the graphics processor for
15 displaying the graphics primitives comprising the display list.

1 2. The system of claim 1 further including a frame buffer coupled
2 to the graphics processor for storing the display list.

1 3. The system of claim 1, wherein the graphics subsystem includes
2 a plurality of storage means for storing the parameter values
3 representative of the graphics primitives in the display list.

1 4. The system of claim 3, wherein the graphics subsystem further
2 includes a address generating means for generating address offsets
3 bits responsive to the display parameter values contained in the
4 display list.

1 5. The system of claim 4, wherein the graphics subsystem further
2 includes an instruction storing means for storing a plurality of
3 instruction data bits responsive to each of the plurality of
4 display list instructions, said instruction data bits shortened to
5 allow the display list to be processed within a shortened
6 processing cycle in the graphics subsystem.

1 6. The system of claim 5, wherein the graphics subsystem further
2 includes an instruction sequencing means coupled to the instruction
3 bits storage means for sequencing bits of the display list into a
4 plurality of register files.

1 7. The system of claim 6, wherein the graphics subsystem further
2 includes an instruction fetch means for randomly fetching a next
3 display list parameter value for a display primitive to be
4 processed in the graphics subsystem.

1 8. The system of claim 7, wherein the graphics subsystem further
2 includes an address counting means for sequentially counting the

3 address storage locations for the display list parameter values of
4 the graphics primitives stored in the plurality of register files.

1 9. The system of claim 8, wherein the graphics subsystem further
2 includes an address partition storage means for storing addresses
3 responsive to load instructions representing the shortened display
4 list, and wherein portions of the instruction storing means index
5 the address partition storage means.

1 10. A graphics system for processing parameter values of graphics
2 primitives in a display list, wherein the display list is shortened
3 to enable fast processing time while maintaining the quality of
4 information contained in the display list, the graphics system
5 comprising:

6 a plurality of register files for storing a plurality of
7 parameter values representing graphics primitives defined in the
8 display list; and

9 a graphics processor coupled to the plurality of register
10 files, wherein the graphics processor processes the shortened
11 display list while maintaining the display quality of primitives
12 displayed in a display unit.

1 11. The graphics processor of claim 10 including an instruction
2 fetch logic unit for fetching the next parameter values responsive
3 to a graphics primitive to be displayed.

1 12. The graphics processor of claim 11 further including a load
2 instruction unit for storing load instructions representative of a
3 shortened display list instruction, said load instruction
4 comprising a plurality of data bits each of said plurality of data
5 bits representing specific load functions to be performed by the
6 load instruction.

1 13. The graphics processor of claim 12, wherein the load
2 instruction unit includes an opcode storage unit for storing opcode
3 information responsive to each of the load instructions for
4 determining the type of function to be performed by the graphics
5 primitive to be rendered.

1 14. The graphics processor of claim 13, wherein the load
2 instruction unit further includes a write enable storage portion
3 for storing write enable data for determining whether to load one
4 of the plurality of registers in the register.

1 15. The graphics processor of claim 14, wherein the load
2 instruction unit further includes an instruction partition portion
3 for storing partition data for referencing the partition table to
4 load parameter values to the referenced register.

1 16. The graphics processor of claim 10 further comprising a data
2 shifter coupled to the load instruction unit for sequentially

3 shifting data bits corresponding to a load instruction in order to
4 write the load instructions to the register files.

1 17. The graphics processor of claim 16 further comprising an
2 address counter coupled to the register files for sequentially
3 counting the address offsets of the register files locations as the
4 load instruction data is loaded into the register files.

1 18. The graphics processor of claim 17 further comprising a
2 partition table coupled to the load instruction unit for storing
3 the address offset bits corresponding to random register locations
4 in the register files for the display parameter values in the
5 display list.

1 19. The graphics processor of claim 18 further comprising a write
2 enable signal coupled to the address bit shifter, said write enable
3 signal asserted high to allow the graphics processor to write the
4 load instructions to the register files, wherein the write enable
5 signal enables the graphics processor to randomly load register
6 locations in the register file.

1 20. The graphics processor of claim 19 further comprising a
2 request next parameter value signal coupled to the fetch logic
3 unit, said request next parameter signal asserted high to allow the
4 next parameter value in the display list to be fetched by the
5 graphics processor.

1 21. The graphics processor of claim 20 wherein the partition look-
2 up table comprises 64 entries of address offsets to the register
3 file.

1 22. The graphics processor of claim 21 wherein the 64 entries of
2 the partition table are evenly distributed to corresponding
3 register locations in the register file.

1 23. The graphics processor of claim 22 wherein each of the 64
2 entries of the partition table is 6 binary wide.

1 24. The graphics processor of claim 23 wherein the register file
2 comprises 1024 entries of addresses.

1 25 A method of encoding and decoding a shortened display list
2 load instruction comprising the steps of:
3 encoding a field load instruction wherein the field load
4 instruction comprises an opcode instruction, a write enable field
5 and a partition index field;
6 loading the field load instruction to a register file; and
7 executing the field load instruction in a shortened processing
8 time.

1 26 The method of claim 25 wherein the field load instruction
2 execution step comprises the step of enabling the write enable

3 field to allow the load instruction to be randomly load to the
4 register file.

1 27. The method of claim 25 wherein the loading step comprises the
2 step of loading a shift register with write enable data from the
3 field load instruction.

1 28. The method of claim 27 wherein the instruction loading step
2 further comprises the step of indexing a partition look-up table
3 with write enable data from the write enable field.

1 29. The method of claim 27 wherein the instruction loading step
2 further includes loading an address counter to sequentially count
3 the number of load instructions in the display list.

1 30. The method of claim 29 wherein the write enable field is
2 disabled to skip the loading of a register in the register file.

ABSTRACT OF THE DISCLOSURE

A graphics system includes a graphics processor for rendering graphics primitives with a shortened display list. A host processor generates a display list which includes a field load instruction for loading the display list into a register file. The graphics processor includes logic to encode and decode the field load instruction thereby shortening the display list loaded into the register file. The field load instruction may also be decoded to allow the graphics processor to randomly load the register file thereby shortening the processing of the display list.

08918053-031497

DECLARATION AND POWER OF ATTORNEY
Original Application

As below named inventor, I declare that I have reviewed and understand the contents of the specification, including the claims, as amended by any amendment specifically referred to in this Declaration, that the information given herein is true, that I believe that I am the original, first and joint inventor of the invention entitled:

METHOD AND APPARATUS FOR SHORTING DISPLAY LIST INSTRUCTIONS

which is described and claimed in:

☒ the attached specification or
☐ the specification in application Serial No. _____ filed _____ amended

that I acknowledge my duty to disclose information in accordance with 37 C.F.R. Section 1.56 and defined on the attached sheet, which is material to the examination of this application, that I do not know and do not believe the same was ever known or used in the United States of America before my or our invention thereof or patented or described in any printed publication in any country before my or our invention thereof, or more than one year prior to this application, or in public use or on sale in the United States of America more than one year prior to this application, that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months prior to this application and that as to applications for patent or inventor's certificate filed by me or my legal representatives or assigns in any country foreign to the United States of America, the earliest filed foreign applications(s) filed within twelve months prior to the filing date of this application and all foreign applications filed more than twelve months prior to the filing date of this application, if any, are identified below.

CHECK APPROPRIATE BOX:

☒ No earlier-filed foreign applications.
☐ Required information as to foreign applications filed prior to filing date of this application is on page 5 attached hereto and made a part hereof.

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

<u>NAME</u>	<u>REGISTRATION NO.</u>	<u>NAME</u>	<u>REGISTRATION NO.</u>
Shirley L. Church	31,858		
Steven A. Shaw	39,368		
Dan Shifrin	34,473		

SEND CORRESPONDENCE TO:

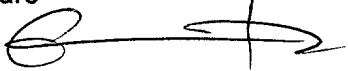
CIRRUS LOGIC, INC.
Legal Department, MS 521
3100 West Warren Avenue
Fremont CA 94538-6423

DIRECT TELEPHONE CALLS TO:

Steven A. Shaw
510-252-6242
Fax: 510-252-6230

(201) FULL NAME OF INVENTOR	LAST NAME DEVIC	FIRST NAME Goran	MIDDLE INITIAL	
RESIDENCE & CITIZENSHIP	CITY Austin	STATE OR FOREIGN COUNTRY Texas	COUNTRY OF CITIZENSHIP US	
POST OFFICE ADDRESS	POST OFFICE ADDRESS 10401 Doc Holliday Trail	CITY Austin	STATE OR COUNTRY TX	ZIP CODE 78753

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Name (201)	Signature	Date
Goran DEVIC		3-7-97

Section 1.56 Duty to Disclose Information Material to Patentability.

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by Sections 1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applications to carefully examine:

(1) prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) the closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made of record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the application takes in:

(i) opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any considerations given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent or inventor.

FIGURE 1

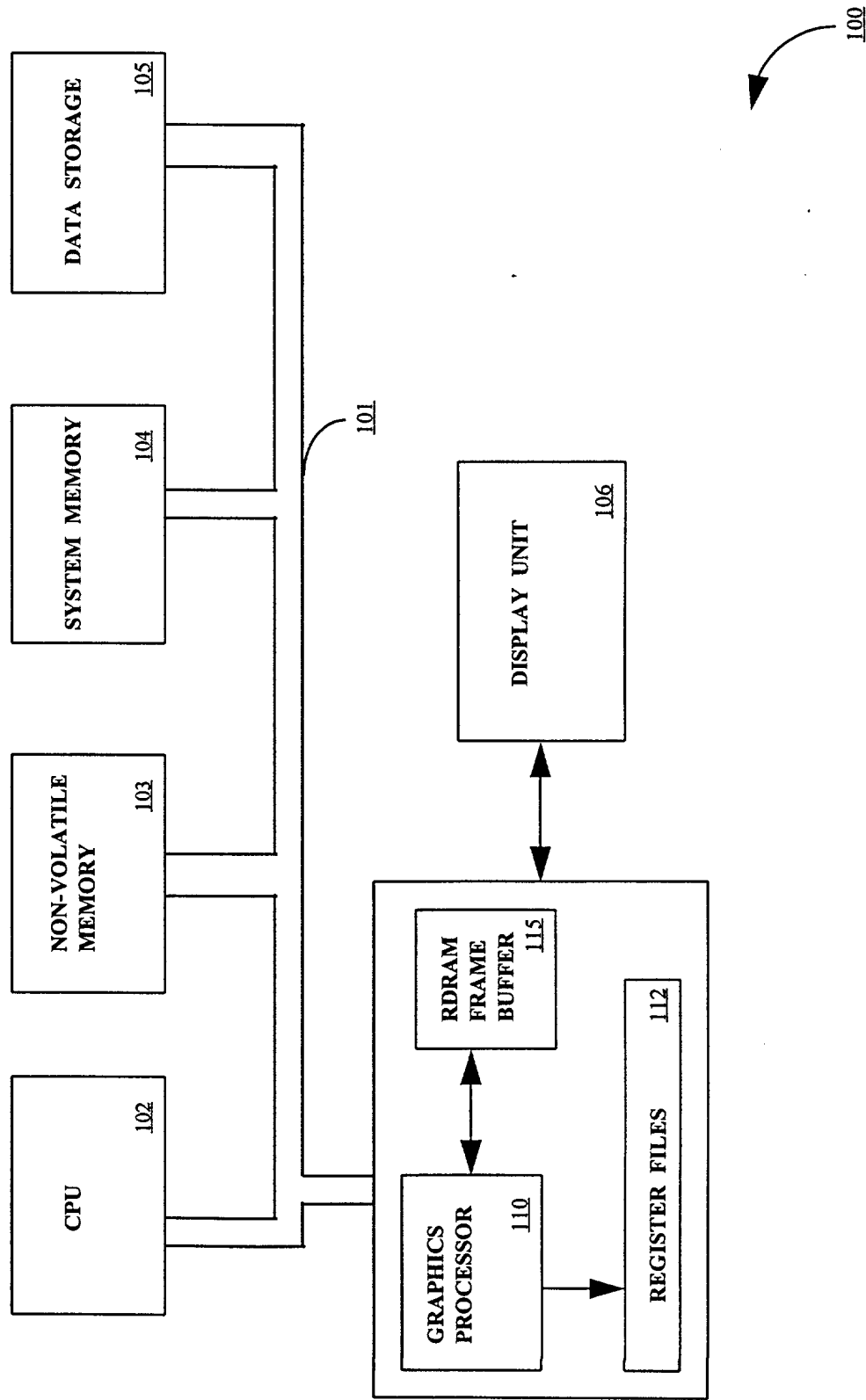


FIGURE 2

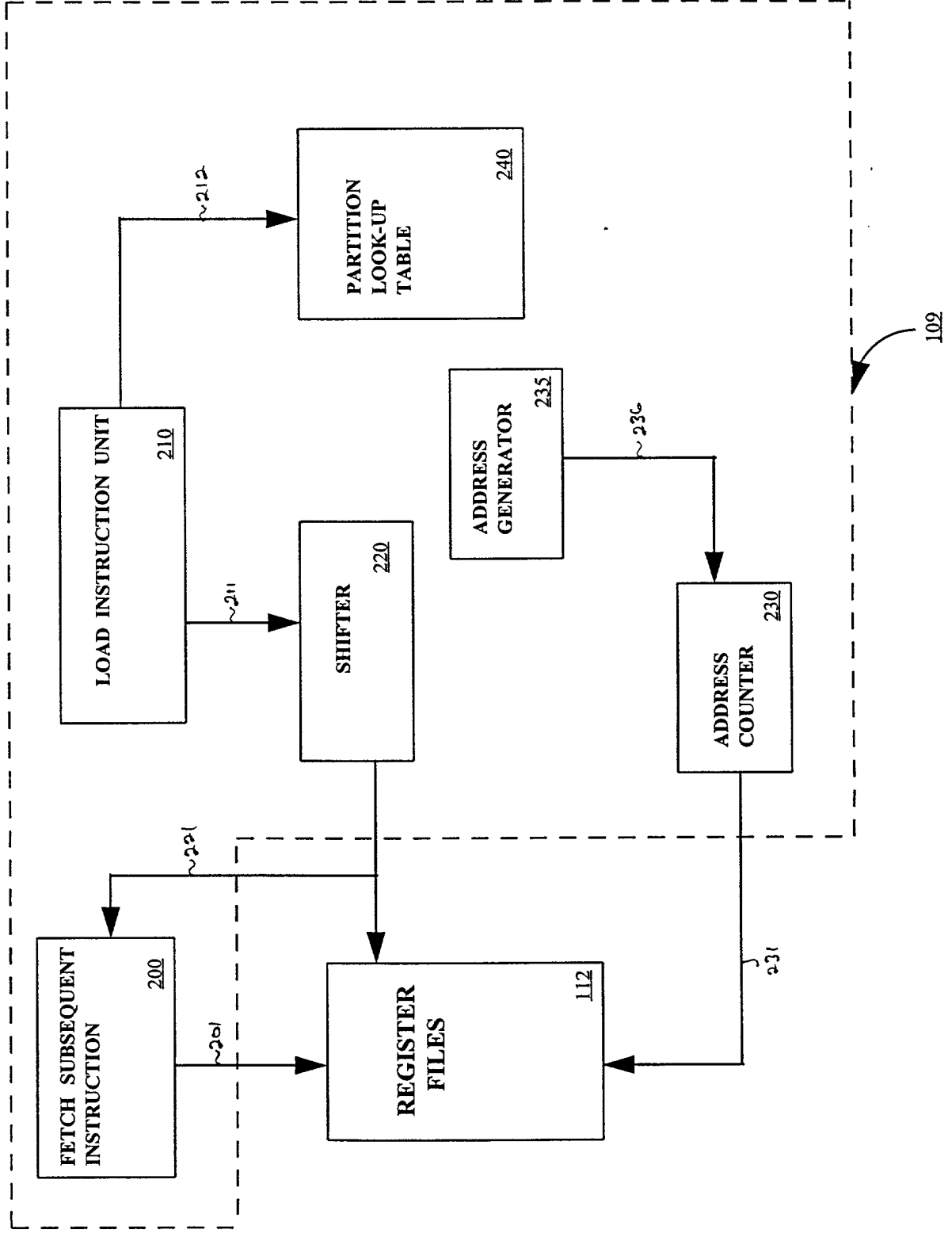


FIGURE 3A

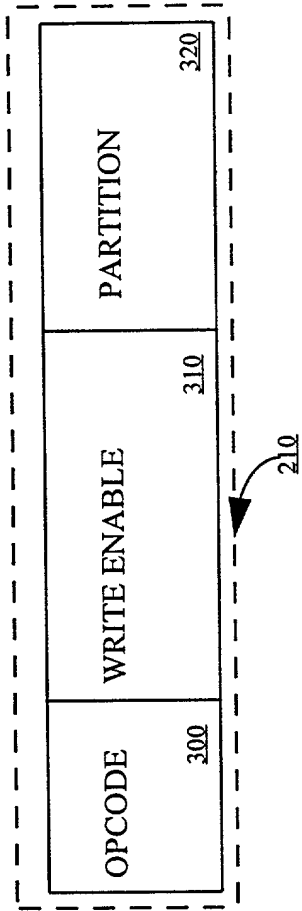


FIGURE 3B

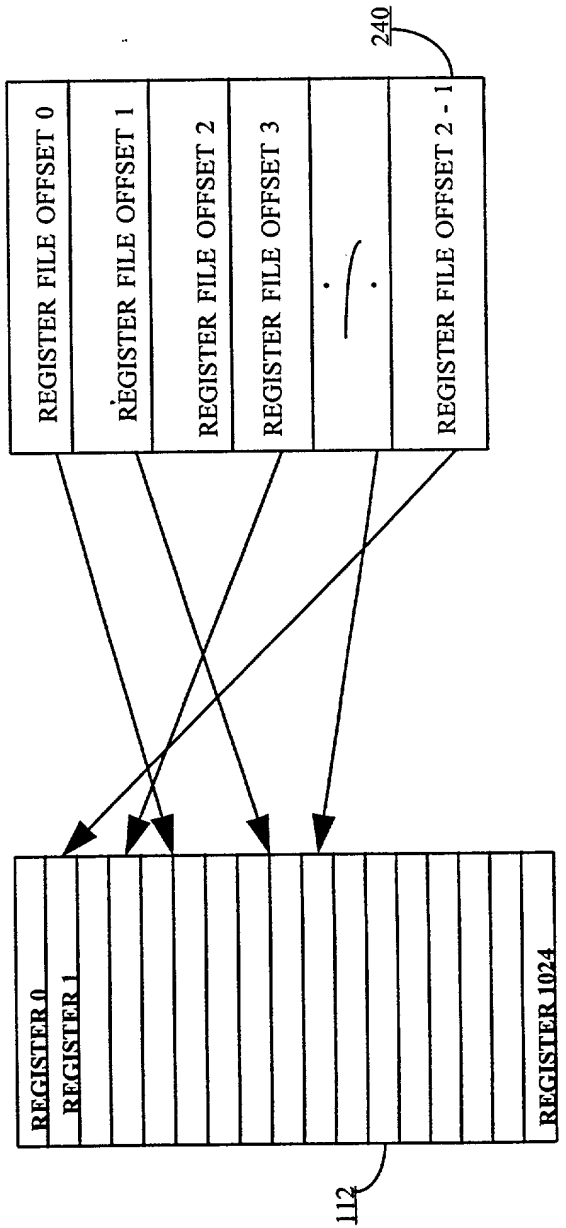


FIGURE 4

